CoSBiLab Graph

User Manual

The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
graph@cosbi.eu

Version 1.0
Date 06/11/2009

You may use, copy, reproduce and distribute this software for any non-commercial purpose, subject to the restrictions in CoSBi-SSLA. This software comes 'as is', with no warranties. This means no express, implied or statutory warranty, including without limitation, warranties of merchantability or fitness for a particular purpose, any warranty against interference with your enjoyment of the software or any warranty of title or noninfringement. There is no warranty that this software will fulfill any of your particular purposes or needs. Also, you must pass this disclaimer on whenever you distribute the software or derivative works.

Contents

1 Introduction **1**
- 1.1 Welcome to CoSBiLab Graph . . . 1
- 1.2 License . . . 1
- 1.3 System requirements . . . 1
- 1.4 Download . . . 1
- 1.5 Support . . . 1

2 Getting Started **1**
- 2.1 Create the first graph . . . 2
- 2.2 Properties Inspector . . . 2
- 2.3 Navigator . . . 3
- 2.4 Items palette . . . 4
- 2.5 Group nodes . . . 4
- 2.6 Import data from Beta Workbench . . . 6

3 Layout **6**

4 Actions **9**
- 4.1 General Informations . . . 10
- 4.2 Components . . . 10
- 4.3 Degree . . . 10
- 4.4 Betweenness Centrality . . . 10
- 4.5 Clustering Coefficient . . . 10
- 4.6 TI,WI . . . 11
  - 4.6.1 TI^n, WI^n . . . 11
  - 4.6.2 TO^n, WO^n . . . 12
- 4.7 Status s, Contrastatus s', Netstatus $\Delta(s)$ . . . 13
- 4.8 K Index . . . 13
- 4.9 Cycles . . . 14
- 4.10 Cycle Sign . . . 14
- 4.11 Shortest Path Matrix . . . 15
- 4.12 Export graph to image . . . 15

5 Advanced Topics **16**
- 5.1 Add Expressions to all nodes or all edges . . . 16
- 5.2 Bind properties with graphical attributes . . . 16

1 Introduction

1.1 Welcome to CoSBiLab Graph

CoSBiLab Graph can construct, visualize and modify graphs as well as calculate measures and layouts. CoSBiLab Graph can import and export data in a variety of formats, among which the reaction network generated by the Beta Workbench[3][2] .

1.2 License

CoSBiLab Graph is a free downloadable application for non commercial purposes covered by *the Microsoft Research - University of Trento Centre for Computational and Systems Biology Shared Source license agreement* (CoSBi-SSLA)

1.3 System requirements

Microsoft .NET Framework 3.5 SP1.

1.4 Download

CoSBiLab Graph is freely downloadable for non commercial purposes at www.cosbi.eu

1.5 Support

CoSBi is very interested in your opinion. If you want to report issues, give feedback or ask a question, send a mail to graph@cosbi.eu

2 Getting Started

Basically there are two options how to start network analysis: creating a graph (File/New graph, see details in 2.1) or importing graph data (File/Open). Importing data is possible in five different formats (including outputs from the BlenX environment, DOT[1] and DL[2]). Imported network data can be changed by adding or removing nodes.

[1]Graph descriptor files supported by Graphviz http://graphviz.org/
[2]DL files are supported by UciNET http://www.analytictech.com/

We will illustrate the most important features of the software by the example of the Kuosheng Bay food web [10], already analysed topologically in details [8]. So, this network of 15 nodes may be regarded as our 'real toy model'.

2.1 Create the first graph

Start CoSBiLab Graph and select *File* → *New Graph...* from the menu or click on the *New* icon showed in the tool bar. A new window, appears to choose the kind of graph you are interested in. By default you can choose only between directed and undirected graph. Click for instance on *Undirected graph*. The main part of the window is now representing a new empty graph called *Graph 1*. The *'insert'* icon is selected in the *'Tools'* tool bar, which means that is possible to add new nodes and links. You can choose the kind of item that you want to add to the graph in the *'Items Palette'* docked window. By default a node is selected. By clicking on the graph area a new node is added. At each new click a new node is added. To add an edge select the edge item from the Items Palette, then press the left button of the mouse over one node in the graph. Keeping the button pressed move the cursor over another node of the graph and release the button. The graph can be saved by clicking on the *'Save'* icon or selecting *File* → *Save* from the menu.

2.2 Properties Inspector

The *Properties Inspector* docked window shows the properties of the selected item and allows you to add/remove or modify properties. A property is any data identify by a unique name inside the item. When performing calculations on the network (see 4), certain outputs (node properties) will become automatically available in the Properties Inspector panel. Then, these node characteristics can be used, for example, for visualisation (see 5.2). The data inside the *Properties Inspector* are referred to the selected item which can be a node, an edge or the whole graph. In order to select a new item click on the *'Select'* icon into the toolbar or select *Tools* → *Select* from the menu. Move the cursor over a node, an edge or the background area for the whole graph and click. Now the *Properties Inspector* appears like in figure 1. At the top of the window is showed the kind of object selected. The main part of

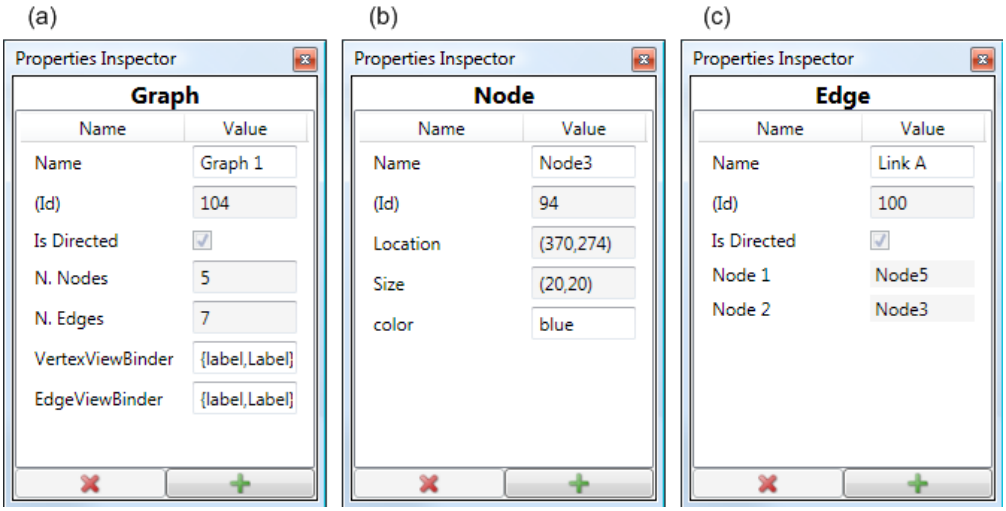


Figure 1: The Properties Inspector showing properties of a graph(a), a node(b) and an edge(c)

the inspector contains a list of property names and values. The background colour of the fields gives you important information about the value:

Gray The value is not editable

White The value can be edited

Yellow The value was just modified

Red The value that you have typed in is not valid and will not be saved

The tool tip which appears by moving the cursor over a value suggests the type of data. At the bottom of the Properties Inspector there are two buttons. The red X remove the selected property if it is not read only. The green 'plus' add a new property, and if clicked, it change the aspect of the *Properties Inspector* like in figure 2. From there you can choose a name, a type and a value for the new property then click again to confirm the addition of the property or click the red X to cancel the operation.

2.3 Navigator

The navigator component shows a map of the entire graph. A dark blue rectangle represent the portion of the graph visualized inside the graph window. The navigator is entirely blue if the complete graph area is visible

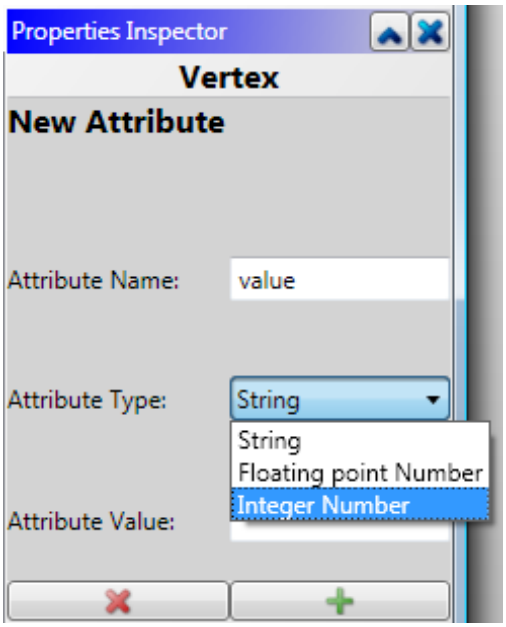


Figure 2: Add a new property

inside the window. By clicking on the map you can move the visualized area to another part of the graph. At the top of the Navigator there are three buttons: Zoom out, Original size and Zoom in which modify the scale rate of the view. Alternatively, when the 'Select' tool is selected, point the mouse over the graph and rotate the wheel of the mouse to zoom. Finally you can also press *Ctrl* and select an area of the graph by keeping pressed the left button of the mouse to zoom in.

2.4 Items palette

The Items palette allows to choose the type of item that will be inserted during the nexts insertion in the graph. Each time that a new item is selected the *'Insert'* Tool is selected so the application is ready to insert that item inside the selected graph. The items inside the items palette depends by the *graph template* selected when the graph was created.

2.5 Group nodes

CoSBiLab Graph can select a group of nodes and collapse them in a single node which mantains the same edges with neighbor nodes. To group nodes, click on the 'group' icon in the tool bar or select *Group* from the *Tools* menu.

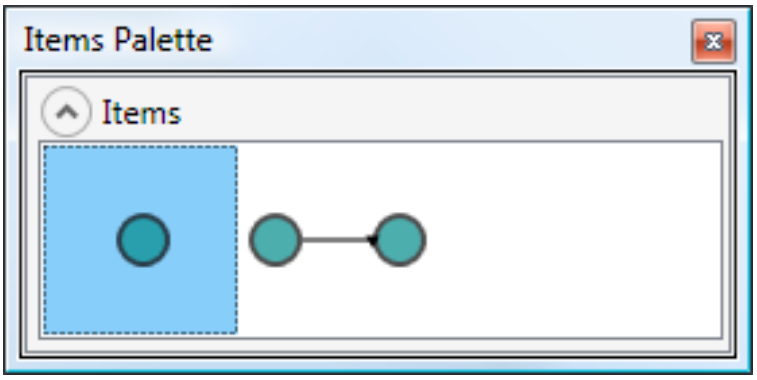


Figure 3: The items palette

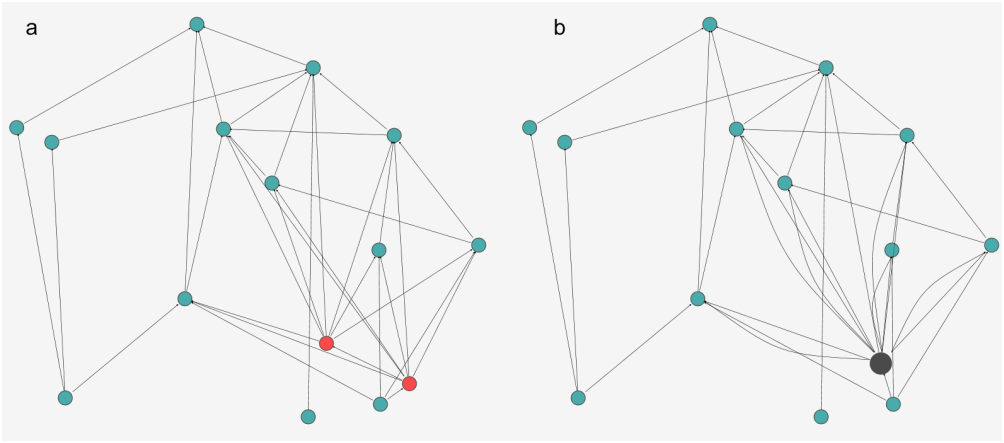


Figure 4: The Kuosheng Bay network before (a) and after (b) aggregating red nodes (the aggregate is the black node).

Move the cursor over the graph, press the left button of the mouse and, by keeping it pressed, move the cursor in order to select an area. The nodes inside the selected area will be collapsed into a single node.
When nodes are grouped, the active tool becomes *'Select'* so that to group other nodes you have to select again the 'group' tool from the tool bar.
Nodes collapsed together can be expanded by right clicking and choosing from the contextual menu the *Expand group* item.

The information regarding the nodes contained inside a cluster node are temporal and are not saved to a file.

Figure 4 shows the original Kuosheng Bay network (Figure 4a) and the same network after aggregating herbivorous zooplankton and carnivorous zooplankton (in red, Figure 4b). Poperties inspector tells how many nodes have been aggregated into particular components. For the aggregates, topological indices can be calculated again after aggregation.

2.6 Import data from Beta Workbench

With CoSBiLab Graph you can extract from a simulation output file (**.spec*) the reaction graph. Click on Open icon or select Open from File menu. Now select BlenX spec files from the file extension combo box at the bottom of the open dialog. Select a spec file and click Open button.
The imported graph represent each reaction as a gray node and each reactants as a colored node with a size proportional to the number of reaction in which it is involved. Another graph generated by the BetaWB is the CTMC which represent the state space of a BlenX program. A Dot file is generated by running the simulator with parameter -r=SBML.

> ***BetaWB*** *is a collection of tools based on the programming language BlenX, explicitly designed to represent biological entities and their interactions. The BetaWB includes the BetaWB simulator, a stochastic simulator based on an efficient variant of the Gillespie Stochastic Simulation Algorithm (SSA), the BetaWB designer, a graphical editor for developing models and the BetaWB plotter, a tool to analyze the results of a stochastic simulation run. For more information please visit* `http://www.cosbi.eu/index.php/research/prototypes/beta-wb`

3 Layout

CoSBiLab Graph implements many layout algorithms to draw a graph. You can choose a layout by selecting an item from the *Layout* menu. Note that some of them cannot be applied to every graph. For example Balloon tree layout can be applied only to directed acyclic graph. When the selected algorithm is not applicable an error message is shown on the console.

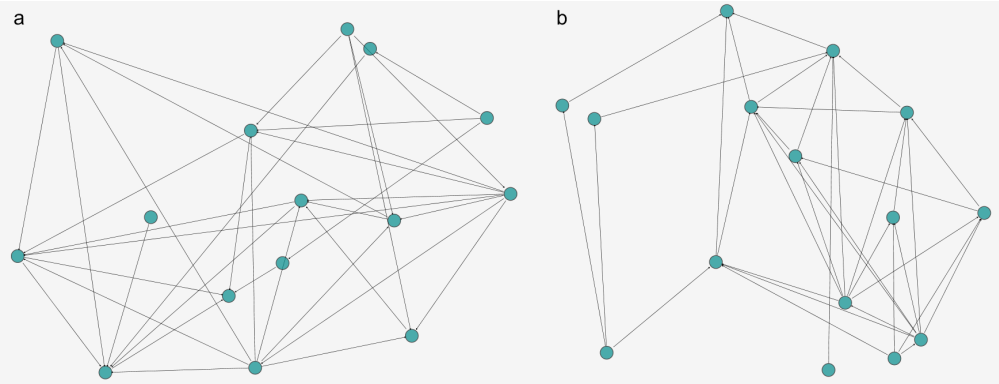
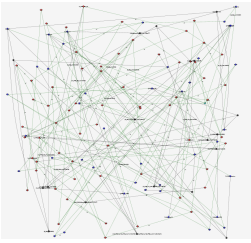
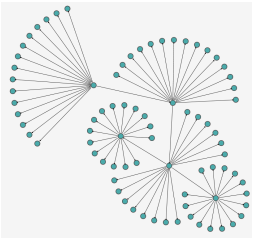
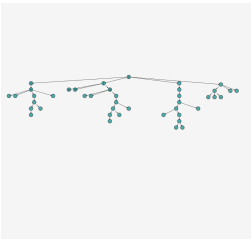
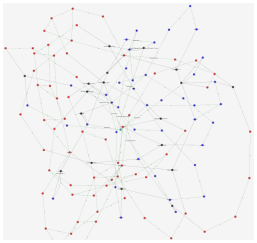
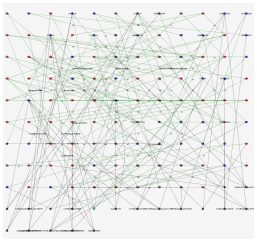
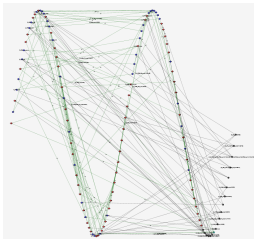
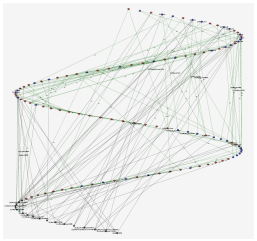
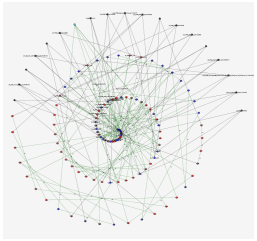


Figure 5: The Kuosheng Bay network in random layout (a) and a manually rearranged, saveable form where plants are at the bottom and top-predators are at the top (b). In the latter, all links point upwards.

Name	**Description**	**Type**	**Example**
Random	Draw the nodes in a random position	any	
Balloon tree	Draw the network according to the graph hierarchy putting the parent at the centre of a circle of children	DAG	
Tree	Draw the graph according its hierarchy putting the children below the parent	Tree	

Name	Description	Type	Example
Fruchterman Reingold	Is a force directed layout [11]	any	
Grid	Draw nodes on the intersections of a grid	any	
H sinuisoid	Draw nodes along a sinusoidal line	any	
V sinuisoid	Draw nodes along a sinusoidal line rotated by 90 degrees	any	
Spiral	Draw nodes along a spiral line	any	

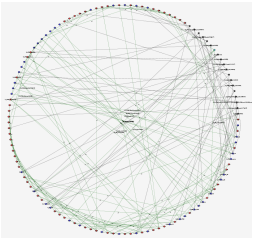
Name	Description	Type	Example
Circle	Draw nodes on a circonference	any	

Table 1: List of layouts

Part of the algorithms derives from NodeXL `http://www.codeplex.com/NodeXL`

It is a comfortable and user-friendly property of the software (compared to several other similar softwares) that rearrangements in graph layout can be saved. Figure 5a shows our food web with a random layout and Figure 5b shows nodes rearranged according to trophic levels (plants are at the bottom and top-predators are at the top). This can be saved and new work can be started from here.

4 Actions

The actions menu offers all the operations and measures computable for the selected graph. Here we provide additional topological indices for network nodes, quantifying various aspects of their indirect neighbourhood in the graph, i.e. characterizing their network position in a more sophisticated way. The menu items with suspension points need some parameters. In this section is gave an overview about the available algorithms.

Some of the actions have a huge computational cost, so run them without tune the parameters or over big graph can freeze the application for a long time.

4.1 General Informations

Graph Statistics print into the console the kind of graph (not directed, directed with cycles, DAG, rooted DAG, forest or tree) and the number of nodes and edges.

4.2 Components

This algorithm count the number of components in the selected graph. A component is a subgraph in which any two nodes are connected to each other by at least a path. When ColorComponents parameter is checked each component will assume a different color.

4.3 Degree

Degree counts the number of incoming (InDegree), ougoing (OutDegree) and adjacent links (Degree) for each node. The last measure is calulated as

$$D_i = D_{in,i} + D_{out,i}$$

In case of undirected graph only the number of adjacent links is reported.

4.4 Betweenness Centrality

Betweenness coefficient quantify the centrality of a node according to the ratio between the number of shortest path passing through that node and the total number of shortest path [1]. Its formulation is

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the number of shortest path between node s and node t and $\sigma_{st}(v)$ is the number of shortest path between node s and node t passing through v. The measure is calculated through the Brandes algorithm [12].

4.5 Clustering Coefficient

The clustering coefficient of a graph node quantifies how densely its neighbours are connected to each other. In other words, it is the number of links between its neighbours divided by the maximum number of links between them. It can be averaged over the whole graph if one wants to provide a global network measure (see [13]).

4.6 TI,WI

The following indices characterize network position in any graphs. The first indices are centrality measures in binary (TI^n) and weighted (WI^n) networks [9], while the indices in the second group are the derived redundancy indices in binary (TO_t^n) and weighted (WO_t^n) networks [8].

4.6.1 TI^n, WI^n

We assume a network with undirected links where effects can spread in any directions. We first consider unweighted networks. Here, we define $a_{n,ij}$ as the effect of j on i when i can be reached from j in n steps. The simplest mode of calculating $a_{n,ij}$ is when n=1 (i.e. the effect of j on i in 1 step): $a_{1,ij} = 1/Di$, where Di is the degree of node i. We assume that indirect effects are multiplicative and additive. For instance, we wish to determine the effect of j on i in 2 steps, and there are two such 2-step pathways from j to i: one is through k and the other is through h. The effects of j on i through k is defined as the product of two direct effects (i.e. $a_{1,kj} \times a_{1,ik}$), therefore the term multiplicative. Similarly, the effect of j on i through h equals to $a_{1,hj,1} \times a_{1,ih}$. To determine the the 2-step effect of j on i ($a_{2,ij}$), we simply sum up those two individual 2-step effects (i.e. $a_{2,ij} = a_{1,kj} \times a_{1,ik} + a_{1,hj} \times a_{1,ih}$) and therefore the term additive.

When effects through n steps are considered, we define the effect received by node i from all other nodes in the same network as:

$$\psi_{n,i} = \sum_{j=1}^{N} a_{n,ij}$$

which is equal to 1 (i.e. each node is affected by the same unit effect). Furthermore, we define the n-step effect originated from node i as:

$$\sigma_{n,i} = \sum_{j=1}^{N} a_{n,ji}$$

that does vary among different nodes (i.e. effects originated from different nodes are generally different). Here, we define the TI^n centrality (TI stands for "topological importance") of node i when effects up to n steps are considered as:

$$TI_i^n = \frac{\sum_{m=1}^{n} \sigma_{m,i}}{n} = \frac{\sum_{m=1}^{n} \sum_{j=1}^{N} a_{m,ji}}{n}$$

that is simply the sum of effects originated from node i up to n steps averaged over by the maximum number of steps considered (n). For weighted networks, all effects are defined in the same way as above but 1-step effects are:

$$a_{1,ij} = \frac{\epsilon_{ij}}{\mu_i}$$

where μ_i is the sum of link weights adjacent to i andϵ_{ij} is the weight of the link between i and j. Furthermore, we define WI_n^i as the topological importance of node i in weighted networks when effects up to n steps are considered:

$$WI_i^n = \frac{\sum_{m=1}^{n} \sigma_{m,i}}{n} = \frac{\sum_{m=1}^{n} \sum_{j=1}^{N} a_{m,ji}}{n}$$

4.6.2 TO^n, WO^n

TI^n (and WI^n) quantifies the relationship between every pairs of nodes in a connected network, if n is large enough (everything is connected to everything else, directly or through indirect pathways). To measure the uniqueness of the position of a network node (to what extent its neighbourhood overlaps with that of others), we define an effect threshold (t) determining the effective range of the interaction structure of node i. We regard nodes within this effective range as strong interactors (the effects they receive from node i is larger than t, while nodes outside this range are weak interactors). Sets of strong interactors of two nodes i and j may overlap. The *topological overlap* (TO_{ij}) between nodes i and j is the number of strong interactors appearing in both i's and j's effective ranges, as a function of maximal step number n and threshold t. The sum of all TO-values between node i and other nodes ($\sum TO_{ij}$ summed over all j with $i \neq j$) provides the summed topological overlap of species i (TO_i). The output of computing the above indices is four matrices (for pairwise TI^n, WI^n, TO_t^n and WO_t^n relationships among nodes) and four vectors for nodal sums. Beyond the exported outputs, the software shows these summed nodal parameters in Properties Inspector.

The parameters are:

Weight attribute name (optional) Set the name of an attribute which indicate the weight of the edge. If the attribute is not present, has an invalid value or this parameter is not set, the weight value is set to 1 by default.

Steps Indicates the number of steps for the index

Compute Up To Step values When checked compute the index considering paths of length less or equal to Steps, otherwise considering paths of length exactly equal to Steps.

Output file name for TI matrix (optional) Set the name of a CSV file in which the TI matrix will be saved

TO Threshold Set the threshold value for TO calculation. If set to 0 the index is not calculated.

4.7 Status s, Contrastatus s', Netstatus $\Delta(s)$

The status of node i(s) in a DAG is the sum of distance from node i to all others. The contra-status (s') is the same if we reverse the direction of all edges in the network. Net status (Δs) equals $\Delta s = s - s'$ [4]. We calculate these indices based on distance (shortest paths; however maximal and average path lengths are also possible). These indices can be calculated only for DAGs, i.e. hierarchies.

4.8 K Index

The keystone index (K) [6] is an ecological adaptation of 'net status' introduced in sociometry [4] and used also in ecology [5]. It can be calculated for DAGs, where each link connects a superior to a subordinate. The keystone index of node i (K_i) is defined as:

$$K_i = K_{bu,i} + K_{td,i} = K_{dir,i} + K_{indir,i} = \sum_{c=1}^{n} \frac{1}{d_c}(1 + K_{bc}) + \sum_{e=1}^{m} \frac{1}{f_e}(1 + K_{te})$$

where n is the number of subordinates of node i, d_c is the number of superiors of its cth subordinate and K_{bc} is the bottom-up keystone index of the c^{th} subordinate. Symmetrically, m is the number of superiors to subordinate i, fe is the number of subordinates of its e^{th} superior and K_{te} is the top-down keystone index of the e^{th} superior. For node i, the first sum in the equation (i.e. $\sum 1/d_c(1 + K_{bc})$) quantifies the bottom-up effect ($K_{bu,i}$) while the second sum (i.e. $\sum 1/f_e(1 + K_{te})$) quantifies the top-down effect ($K_{td,i}$). After rearranging the equation, terms including K_{bc} and K_{te} (i.e. $\sum K_{bc}/d_c + \sum K_{te}/f_e$) refer to indirect effects for node i ($K_{indir,i}$), while terms not containing K_{bc} and K_{te} (i.e. $\sum 1/d_c + \sum 1/f_e$) refer to direct ones ($K_{dir,i}$). Both $K_{bu,i} + K_{td,i}$ and $K_{indir,i} + K_{dir,i}$ equals K_i.

K quantifies only vertical indirect relationships but characterises positional importance by separating indirect from direct, as well as bottom-up from top-down effects in the network. Its important feature is the sensitivity to both distance and degree: it quantifies meso-scale network position [7]. These indices can be calculated only for DAGs, i.e. hierarchies.

4.9 Cycles

This algorithm calculate the number of different cycles belonging to the graph. It can also enumerate them and save the list in a CSV file. The number of cycles is displayed after the execution in the console. The parameters are:

Threshold Set the maximum length for recognized cycles. When its value is set to 0 means that there is no limitation. When this algorithm is executed over a huge graph is recommendable set a threshold different than 0.

OutputFile (optional) Set the path of an output file where all the cycles detected are listed.

AllowVerticesRepetition When checked means that a cycle can pass infinity times by the same node. When checked threshold must be set different than 0.

4.10 Cycle Sign

This algorithm, similarly to the cycles algorithm, enumerate all the cycles in a graph and for each of them calculate a sign. At the end of the execution a summary table will appears into the console. For each cycle length there is a row with the length, the number of cycles of that length, percentage of positive and negative cycles, mean weight, mean weight in positive loops and mean weight in negative loops The parameters are:

SignAttribute Set the name of the sign attribute in the edges. It can be a number or a string which assume values "+" or "-". If an edge don't have this attribute plus sign is assumed by default.

WeightAttribute (optional) Set the name of the weight attribute in the edges. This attribute can assume numeric values. If an edge don't have this attribute a value of 1 is assumed by default.

Threshold Set the maximum length for recognized cycles. When its value is set to 0 means that there is no limitation. When this algorithm is executed over a huge graph is recommendable set a threshold different than 0.

OutputFile (optional) Set the path of an output file where all the cycles detected are listed with their sign and weight.

AllowVerticesRepetition When checked means that a cycle can pass infinity times by the same node. When checked threshold must be set different than 0.

The ratio of negative cycles to total cycles provides a measure of structural balance [5].

4.11 Shortest Path Matrix

Computes the shortest path for each pair of nodes and print the results in a matrix. When there are no path between two nodes the distance is set to 0. The parameters are:

Inverse Path When checked in a directed graph compute the shortest path walking through the edges in the opposite direction

Weight attribute (optional) Specify a name of an edge attribute which represent the wheight.

Output File Name Specify a name for a CSV file in which the resulting matrix will be printed

4.12 Export graph to image

Export to a PNG file the current view of the selected graph. The parameters are:

File Name Specify the output file name

DPI Specify the Dot Per Inch resolution of the output file

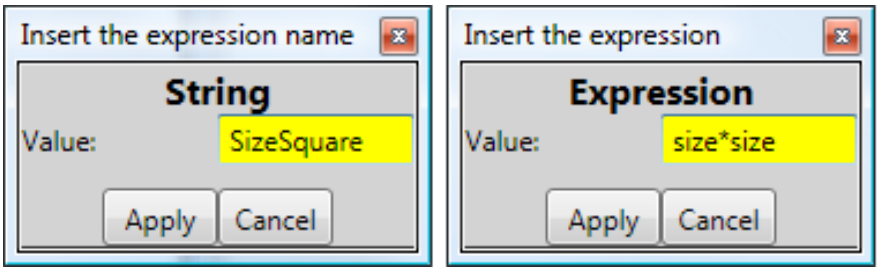


Figure 6: Add expression windows

5 Advanced Topics

5.1 Add Expressions to all nodes or all edges

You can add a mathematical expression to all nodes or all edges by selecting *Add expression to all nodes* or *Add expression to all edges* from the *Graph* menu. In the input box type a name to identify the new expression, then click the apply button. At this point a new input box in which you can specify the expression appears. The expression can be composed by numbers, arithmetical operators (+,-,*,/), parenthesis and the mathematical functions pow(base,exp) and sqrt(number). Instead of a number you can put the name of a property belonging to a node/edge (e.g. $(value + 10)/5$). If the expression is correct, it will have a yellow background, otherwise red. When the expression is ready click apply to copy and evaluate it in each node/edge. If the expression cannot be evaluated in a node or in an edge because the properties used inside the expression are not present or because their values are not numbers, the expression appears as its formulation surrounded by square parenthesis, otherwise appears its evaluation. Note that if the expression cannot be evaluated due to values out of expected range (e.g. sqrt(-5)) the evaluation will be *NaN* that means not a number.

5.2 Bind properties with graphical attributes

The graph visualization can become a lot more intuitive and expressive if properties belonging to nodes and edges are represented through graphical attributes like color, size, etc. CoSBiLab Graph allows you to do it easily.
With the *Select* tool activated, click on an empty area of the graph in order to visualize the graph properties inside the *Properties Inspector*. You can note two properties named VertexViewBinder and EdgeViewBinder. These

two properties are lists of element of the format

$$\{PropertyName, GraphicalAttribute\}$$

Each of these elements bind a property to a graphical element. The type and the value of the property must be compatible with the graphical attribute otherwise the binding has no effect.
Let's introduce an example. You can bind a property called *attribute1* with the graphical attribute *Width* by appending the string {attribute1,Width} to the VertexViewBinder. If none of your nodes has a property called attribute1 you will not note any difference. Now select a node and through the Properties Inspector add a new Property called attribute1, of type String and with value 40. As soon as you click on the *Add* button the selected node will double its width. If you try to play a bit with the value of attribute1 you will note that at each change in the value of attribute1 the size of the node is modified. Now try to change the value of attribute1 to "red". The width of the node is reset to its default value because the system cannot bind a string value like "red" to a numeric attribute like Width. Same thing happen when the value is out of range, as setting a value less than 20 for attribute1.
In another example (figure 7) is showed how the graphical attributes are used at the same time to represent different indexes, in particular the size of the nodes is proportional to TI^3 index and the color with K. Table 2 contains all the graphical attributes that can be set for a node. Table 3 contains all the graphical attributes that can be set for an edge.

XAML *is a declarative XML-based language which is used to initialize structured values and objects. For more information please visit* `http://msdn.microsoft.com/en-us/library/ms747122.aspx`

An interesting feature of the visualization toolkit is to use pictures for representing graph nodes (e.g. a lion). It can be used for education purposes or just simply to improve the didactics of professional presentations.

Pay attention to letter case of properties and graphical attribute, they are case sensitive. That means for example that Label (with capital L) is different than label (with small l).

Name	Description	Accepted values
Label	Set a string that will appear over a node	any value
Color	Set the fill color of a node	names of colors or numbers between 0 and 100 that will be represented as a gray scale
BorderColor	Set the outline color of a node	names of colors
BorderSize	Set the thickness of the node outline	any number
Width	Set the width of a node	numbers greater than 20
Height	Set the height of a node	numbers greater than 20
GraphicElement	Set the node aspect	a string containing XAML code
GraphicElementFile	Set the node aspect	a path of a file which contains XAML code

Table 2: Graphical attributes of a node

Name	Description	Accepted values
Label	Set a string that will appear over an edge	any value
Color	Set the color of an edge	names of colors or numbers between 0 and 100 that will be represented as a gray scale
StrokeThickness	Set the thickness of an edge	any number

Table 3: Graphical attributes of an edge

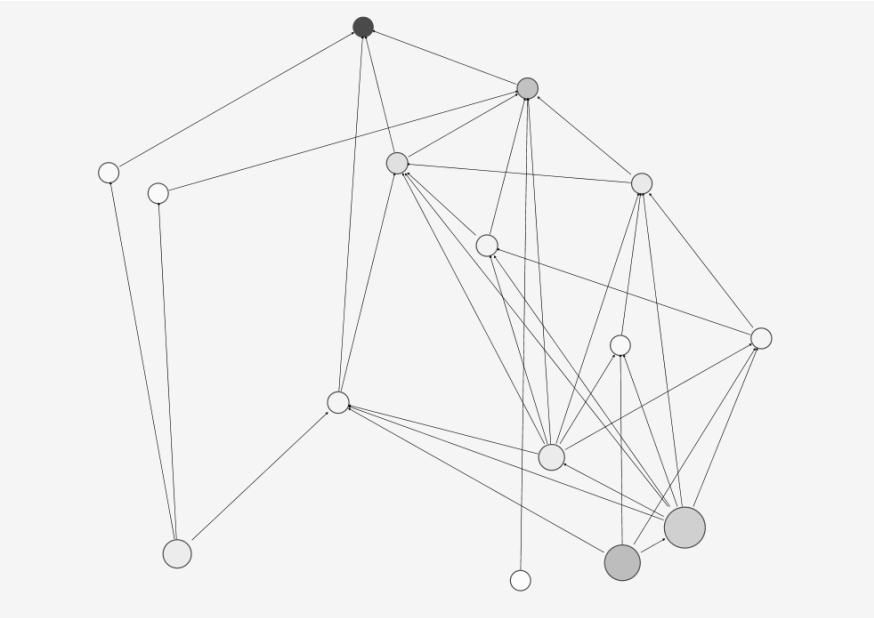


Figure 7: Indexes represented by size and color: the size of the nodes is proportional to TI^3 index and the darkness with K

References

[1] Freeman L C. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[2] Lorenzo Dematté, Corrado Priami, and Alessandro Romanel. The beta workbench: a computational tool to study the dynamics of biological systems. *Briefings in Bioinformatics*, 9:437 – 449, 2008.

[3] Lorenzo Dematté, Corrado Priami, and Alessandro Romanel. The blenx language: A tutorial. In LNCS, editor, *SFM 2008*, pages 313–365. Springer-Verlag, 2008.

[4] Harary F. Status and contrastatus. *Sociometry*, 22:23 – 43, 1959.

[5] Harary F. A structural analysis of the situation in the middle east in 1956. *The Journal of Conflict Resolution*, 5:167 – 178, 1961.

[6] Jordan F., Takacs-Santa A., and Molnar I. A reliability theoretical quest for keystones. *Oikos*, 86:453 – 462, 1999.

[7] Jordan F and Scheuring I. Searching for keystones in ecological networks. *Oikos*, 99:607 – 612, 2002.

[8] Jordan F., Liu W.-C., and Mike A. Trophic field overlap: a new approach to quantify keystone species. *Ecological Modelling*, 220:2899 – 2907, 2009.

[9] Jordan F., Liu W.-C., and van Veen F.J.F. Quantifying the importance of species and their interactions in a host-parasitoid community. *Community Ecology*, 4:79 – 88, 2003.

[10] Lin H-J., Shao K-T., Hwang J-S., Lo W-T., Cheng I-J., and Lee L-H. A trophic model for kuosheng bay in northern taiwan. *J. Mar. Sci. Technol.*, 12:424 – 442, 2004.

[11] Fruchterman T and Reingold E. Graph drawing by force-directed placement. *Software Practice and Experience*, 21:1129–1164, 1991.

[12] Brandes U. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[13] Duncan J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440 – 442, 1998.